CSE211Digital Design

Akdeniz University

Week04-05: Boolean Algebra and Logic Gates

Assoc.Prof.Dr. Taner Danışman
tdanisman@akdeniz.edu.tr

Course program

Week 01	19-Sep-22	Introduction
Week 02	26-Sep-22	Digital Systems and Binary Numbers I
Week 03	03-Oct-22	Digital Systems and Binary Numbers II
Week 04	10-Oct-22	Boolean Algebra and Logic Gates I
Week 05	17-Oct-22	Boolean Algebra and Logic Gates II
Week 06	24-Oct-22	Boolean Algebra and Logic Gates III
Week 07	31-Oct-22	Gate Level Minimization
Week 08	07-Nov-22	Midterm
Week 09	14-Nov-22	Karnaugh Maps
Week 10	21-Nov-22	Karnaugh Maps
Week 11	28-Nov-22	Combinational Logic
Week 12	05-Dec-22	Combinational Logic
Week 13	12-Dec-22	Timing, delays and hazards
Week 14	19-Dec-22	Synchronous Sequential Logic
Week 15	26-Dec-22	Arduino Programming

Previous chapter terms to know

- DVD
- GUI
- BCD
- ASCII
- STX
- ETX
- CR
- HDL

Boolean Algebra

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.
  - In formal logic, these values are "true" and "false."
  - In digital systems, these values are "on" and "off," 1 and 0, or "high" and "low."
- Boolean expressions are created by performing operations on Boolean variables.
  - Common Boolean operators include AND, OR, and NOT.

Boolean Algebra – Truth Table

- A Boolean operator can be completely described using a truth table.
- The truth table for the Boolean operators AND and OR are shown at the right.
- The AND operator is also known as a Boolean product. The OR operator is the Boolean sum.

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Algebra – Truth Table

- The truth table for the Boolean NOT operator is shown at the right.
- The NOT operation is most often designated by an overbar. It is sometimes indicated by a prime mark (‘) or an “elbow” (¬).

NOT x

x	$\overline{x}$
0	1
1	0

Boolean Algebra – Boolean Functions

- A Boolean function has:
  - At least one Boolean variable,
  - At least one Boolean operator, and
  - At least one input from the set {0,1}.
- It produces an output that is also a member of the set {0,1}.

Now you know why the binary numbering system is so handy in digital systems.

Boolean Algebra – Boolean Functions

- The truth table for the Boolean function:

  $$F(x,y,z) = x\bar{z}+y$$

  is shown at the right.
- To make evaluation of the Boolean function easier, the truth table contains extra (shaded) columns to hold evaluations of subparts of the function.

$$F(x,y,z) = x\bar{z}+y$$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Boolean Algebra – Rules of precedence

- As with common arithmetic, Boolean operations have **rules of precedence**
- The **NOT** operator has highest priority, followed by AND and then OR.
- This is how we chose the (shaded) function subparts in our table.

$$F(x,y,z) = x\bar{z}+y$$

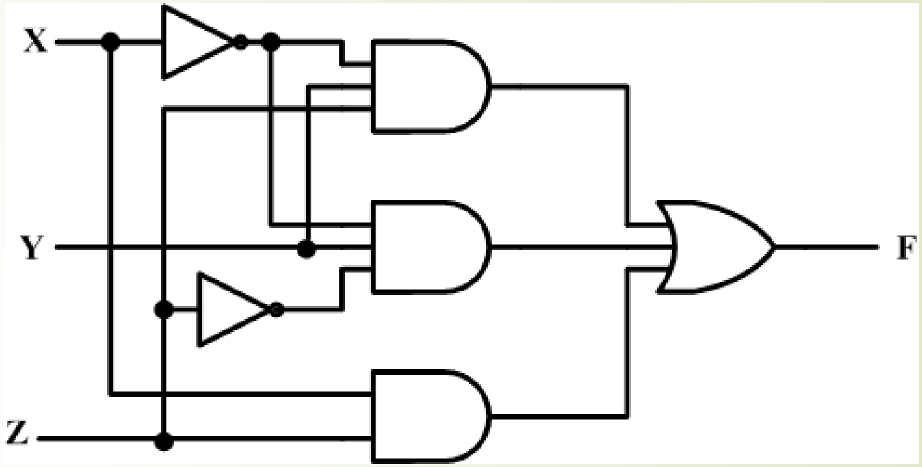
x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Boolean Algebra – Simplification

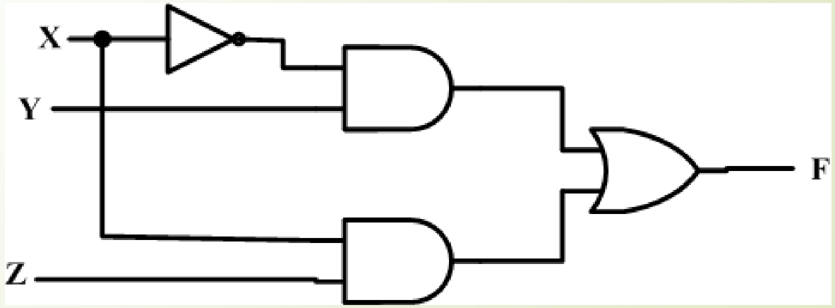
- Digital computers contain circuits that implement Boolean functions.
- The simpler that we can make a Boolean function, the smaller the circuit that will result.
  - Simpler circuits are cheaper to build, consume less power, and run faster than complex circuits.
- With this in mind, we always want to reduce our Boolean functions to their simplest form.
- There are a number of Boolean identities that help us to do this.

Boolean Algebra –Affect on implementation

- F = X'YZ + X'YZ' + XZ



- Reduces to F = X'Y + XZ



Boolean Algebra – Laws

- Most Boolean identities have an AND (product) form as well as an OR (sum) form. We give our identities using both forms. Our first group is rather intuitive:

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$x\overline{x} = 0$	$x + \overline{x} = 1$

Boolean Algebra – Laws (Cont.)

- Our second group of Boolean identities should be familiar to you from your study of algebra:

Identity Name	AND Form	OR Form
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x + (y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$

- Using **distributive law** $x+x'y=(x+x')(x+y)$

Boolean Algebra – Laws (Cont.)

- Our last group of Boolean identities are perhaps the most useful.
- If you have studied set theory or formal logic, these laws are also familiar to you.

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \overline{x} + \overline{y}$	$\overline{(x+y)} = \overline{x}\,\overline{y}$
Double Complement Law	$\overline{(\overline{x})} = x$	

Boolean Algebra – DeMorgan's Law

- Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly.
- DeMorgan's law provides an easy way of finding the complement of a Boolean function.
- Recall **DeMorgan's law** states:

$$\overline{(xy)} = \overline{x} + \overline{y} \quad \text{and} \quad \overline{(x+y)} = \overline{x}\,\overline{y}$$

Boolean Algebra – Proof: DeMorgan

Theorem 3.7: DeMorgan's Law

For each pair of elements x, y in a Boolean algebra:

$$\overline{(x + y)} = \overline{x} \cdot \overline{y}$$

$$\overline{(x \cdot y)} = \overline{x} + \overline{y}$$

Proof:

$$(x + y) + \overline{x} \cdot \overline{y} = (x + y + \overline{x}) \cdot (x + y + \overline{y})$$

$$= (x + \overline{x} + y) \cdot (y + \overline{y} + x)$$

$$= (1 + y) \cdot (1 + x)$$

$$= 1 \cdot 1$$

$$= 1$$

Note: Using Associativity of each operation (+), (·)

Boolean Algebra – DeMorgan's Law (Cont.)

- DeMorgan's law can be extended to any number of variables.
- Replace each variable by its complement and change all ANDs to ORs and all ORs to ANDs.
- Thus, we find the complement of:

$$F(X,Y,Z) = (XY) + (\overline{X}Z) + (Y\overline{Z})$$

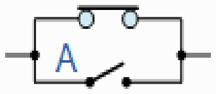
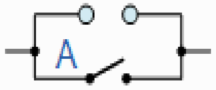
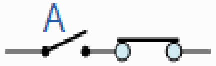
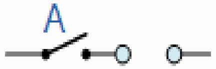
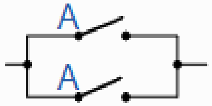
is:

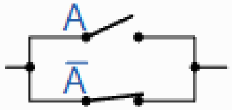
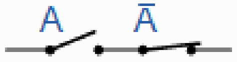
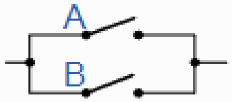
$$\overline{F}(X,Y,Z) = \overline{(XY) + (\overline{X}Z) + (Y\overline{Z})}$$

$$= \overline{(XY)}\,\overline{(\overline{X}Z)}\,\overline{(Y\overline{Z})}$$

$$= (\overline{X}+\overline{Y})(X+\overline{Z})(\overline{Y}+Z)$$

Truth Tables for the Laws of Boolean

Boolean Expression	Description	Equivalent Switching Circuit	Boolean Algebra Law or Rule
A + 1 = 1	A in parallel with closed = "CLOSED"		Annulment
A + 0 = A	A in parallel with open = "A"		Identity
A . 1 = A	A in series with closed = "A"		Identity
A . 0 = 0	A in series with open = "OPEN"		Annulment
A + A = A	A in parallel with A = "A"		Idempotent
A . A = A	A in series with A = "A"		Idempotent

Boolean Expression	Description	Equivalent Switching Circuit	Boolean Algebra Law or Rule
$NOT\ \overline{A} = A$	NOT NOT A (double negative) = "A"		Double Negation
$A + \overline{A} = 1$	A in parallel with NOT A = "CLOSED"		Complement
$A.\overline{A} = 0$	A in series with NOT A = "OPEN"		Complement
$A{+}B = B{+}A$	A in parallel with B = B in parallel with A		Commutative
$A.B = B.A$	A in series with B = B in series with A		Commutative
$\overline{(A{+}B)} = \overline{A}.\overline{B}$	invert and replace OR with AND		de Morgan's Theorem
$\overline{(A.B)} = \overline{A} + \overline{B}$	invert and replace AND with OR		de Morgan's Theorem

Boolean Algebra – Complement of Functions

EXAMPLE 2.2

Find the complement of the functions $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$. By applying DeMorgan's theorems as many times as necessary, the complements are obtained as follows:

$$F_1' = (x'yz' + x'y'z)' =$$

$$F_2' = [x(y'z' + yz)]' =$$

$$=$$

$$=$$

Boolean Algebra – Principle of Duality

- What is meant by the dual of a function?
  - The *dual* of a function is obtained by interchanging OR and AND operations and replacing 1s and 0s with 0s and 1s.
- Shortcut to getting function complement
  - Having F=X'YZ'+X'Y'Z
  - Generate the dual F=(X'+Y+Z')(X'+Y'+Z)
  - Complement each literal to get:
  - F'=(X+Y'+Z)(X+Y+Z')
- Each postulate consists of two expressions s.t. one expression is transformed into the other by interchanging the operations (+) and (·) as well as the identity elements 0 and 1.
- Such expressions are known as duals of each other.
- If some equivalence is proved, then its dual is also immediately true.
- E.g. If we prove: $(x \cdot x) + (\overline{x} \cdot \overline{x}) = 1$, then we have by duality: $(x + x) \cdot (\overline{x} + \overline{x}) = 0$

Boolean Algebra – Principle of Duality (Cont.)

- There are useful identities of Boolean expressions that can help us to transform an expression A into an equivalent expression B
- We can derive additional identities with the help of the **dual** of a Boolean expression.
- The dual of a Boolean expression is obtained by interchanging Boolean sums and Boolean products and interchanging 0s and 1s.

Boolean Algebra – Duality

- **Examples:**

The dual of $x(y + z)$ is $x + yz$

The dual of $x \cdot 1 + (y + z)$ is $(x + 0)(yz)$

The **dual** of a Boolean function F represented by a Boolean expression is the function represented by the dual of this expression.

This dual function, denoted by Fd, does not depend on the particular Boolean expression used to represent F.

Boolean Algebra – Duality

- Therefore, an identity between functions represented by Boolean expressions **remains valid** when the duals of both sides of the identity are taken.
- We can use this fact, called the **duality principle**, to derive new identities.
- For example, consider the absorption law $x(x + y) = x$
- By taking the duals of both sides of this identity, we obtain the equation $x + xy = x$, which is also an identity (and also called an absorption law).

Boolean Algebra – Example

- We can use Boolean identities to simplify the function: $F(X,Y,Z) = (X + Y)(X + \overline{Y})\overline{(X\overline{Z})}$ as follows:

$(X + Y)(X + \overline{Y})\overline{(X\overline{Z})}$	Idempotent Law (Rewriting)
$(X + Y)(X + \overline{Y})(\overline{X} + Z)$	DeMorgan's Law
$(XX + X\overline{Y} + XY + Y\overline{Y})(\overline{X} + Z)$	Distributive Law
$((X + Y\overline{Y}) + X(Y + \overline{Y}))(\overline{X} + Z)$	Commutative & Distributive Laws
$((X + 0) + X(1))(\overline{X} + Z)$	Inverse Law
$X(\overline{X} + Z)$	Idempotent Law
$X\overline{X} + XZ$	Distributive Law
$0 + XZ$	Inverse Law
XZ	Idempotent Law

Boolean Algebra Simplification Example

Q =	(A + B).(A + C)	
	A.A + A.C + A.B + B.C	– Distributive law
	A + A.C + A.B + B.C	– Idempotent AND law (A.A = A)
	A(1 + C) + A.B + B.C	– Distributive law
	A.1 + A.B + B.C	– Identity OR law (1 + C = 1)
	A(1 + B) + B.C	– Distributive law
	A.1 + B.C	– Identity OR law (1 + B = 1)
Q =	A + (B.C)	– Identity AND law (A.1 = A)

Boolean Algebra – Summary

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Boolean Algebra – Simplification

Simplify the following Boolean functions to a minimum number of literals.

1. $x(x' + y) =$
2. $x + x'y =$
3. $(x + y)(x + y') =$
4. $xy + x'z + yz =$

   $=$

   $=$

   $=$
5. $(x + y)(x' + z)(y + z) =$.

Boolean Algebra – Simplification

- Find the simplified equivalent for the following equations
- **a(a+b')**
  -
- **a+(a'+b)'**
  -
- **x.y+x'.z+y.z+y'z**
  -
- xyz+x'yz+y'z
  -

Boolean Algebra – Simplification

- $A.(A'+B)=?$
- $(A+B).(A+C)=?$
- $F=A'.B+A+A.B=?$
- $A.B+A'C+B.C$
- $A'B'C' + A'B'C + ABC' + AB'C'$

Boolean Algebra – Simplification

$F(A,B,C) = A'B'C' + A'B'C + AB'C' + AB'C + ABC' + ABC$

=

=

=

=

=

Boolean Algebra – Simplification

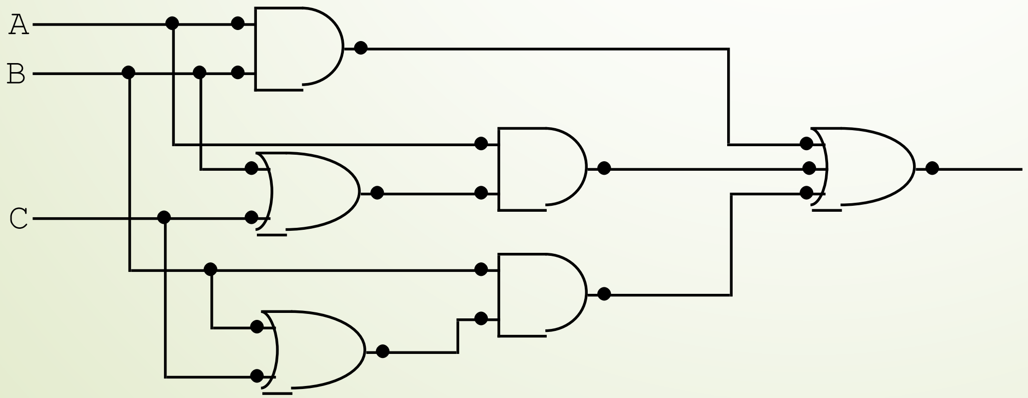
- **AB + A(B + C) + B(B + C)**

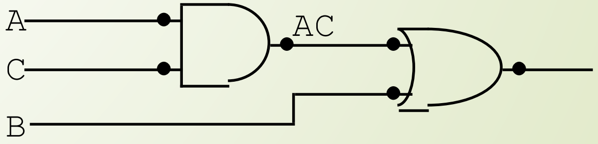
= AB + AB + AC + BB + BC

= AB + AC + B + BC

= AB + AC + B

=AC + B





Boolean Algebra – Logically Equivalent Expressions

- Through our exercises in simplifying Boolean expressions, we see that there are numerous ways of stating the same Boolean expression.
  - These "synonymous" forms are ***logically equivalent***.
  - **Logically equivalent expressions have identical truth tables.**
- In order to eliminate as much confusion as possible, designers express Boolean functions in ***standardized*** or ***canonical*** form.

Boolean Algebra – Sum of Products and Product of Sums

- There are two canonical forms for Boolean expressions: **sum-of-products** and **product-of-sums**.
  - Recall the Boolean product is the AND operation and the Boolean sum is the OR operation.
- In the **sum-of-products** form, ANDed variables are ORed together.
  - For example: $F(x,y,z) = xy + xz + yz$
- In the **product-of-sums** form, ORed variables are ANDed together:
  - For example: $F(x,y,z) = (x+y)(x+z)(y+z)$

Boolean Algebra – Normal Forms

- Consider the function:

  $$f(w,x,y,z) = \bar{x} + w\bar{y} + \bar{w}\bar{y}z$$

- A **literal** is an occurrence of a complemented or uncomplemented variable in a formula.
- A **product term** is either a literal or a product (conjunction) of literals.
- **Disjunctive normal form**: A Boolean formula written as a single product term or as a sum (disjunction) of product terms.

Boolean Algebra – Normal Forms

- Consider the function:

$$f(w, x, y, z) = z(x + \overline{y})(w + \overline{x} + \overline{y})$$

- A **sum term** is either a literal or a sum (disjunction) of literals.
- **Conjunctive normal form**: A Boolean formula written as a single sum term or as a product (conjunction) of sum terms.

Boolean Algebra – Canonical Formulas

- How to obtain a Boolean formula given a truth table?

X	Y	Z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Boolean Algebra – Minterm Canonical Formula

X	Y	Z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$\bar{x}\,\bar{y}\,z$

$\bar{x}\,y\,z$

$x\,\bar{y}\,\bar{z}$

$$f(x,y,z) = \bar{x}\,\bar{y}\,z + \bar{x}y\,z + x\,\bar{y}\,\bar{z}$$

Boolean Algebra – m-Notation

X	Y	Z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$\bar{x}\,\bar{y}\,z$

$\bar{x}\,y\,z$

$x\,\bar{y}\,\bar{z}$

- $f(x,y,z)$ can be written as $f(x,y,z) = m_1 + m_3 + m_4$
- $f(x,y,z) = \Sigma m(1,3,4)$

Boolean Algebra – Maxterm Canonical Formula

X	Y	Z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$x + y + z$

$x + \overline{y} + z$

$\overline{x} + y + \overline{z}$

$\overline{x} + \overline{y} + z$

$\overline{x} + \overline{y} + \overline{z}$

$$f(x,y,z) = (x + y + z)(x + \overline{y} + z)(\overline{x} + y + \overline{z})(\overline{x} + \overline{y} + z)(\overline{x} + \overline{y} + \overline{z})$$

Boolean Algebra – Sum of Products

- It is easy to convert a function to sum-of-products form using its truth table.
- We are interested in the values of the variables that make the function true (=1).
- Using the truth table, we list the values of the variables that result in a true function value.
- Each group of variables is then ORed together.

$F(x,y,z) = x\bar{z}+y$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Boolean Algebra – Sum of Products

- The sum-of-products form for our function is:

$$F(x,y,z) = \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}\bar{z} + xy\bar{z} + xyz$$

We note that this function is not in simplest terms. Our aim is only to rewrite our function in canonical sum-of-products form.

$$F(x,y,z) = x\bar{z} + y$$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

CNF and DNF

- Find the *f* function using both DNF and CNF for the following truth tables

X	Y	*f*
0	0	0
0	1	1
1	0	1
1	1	0

X	Y	*f*
0	0	1
0	1	1
1	0	1
1	1	1

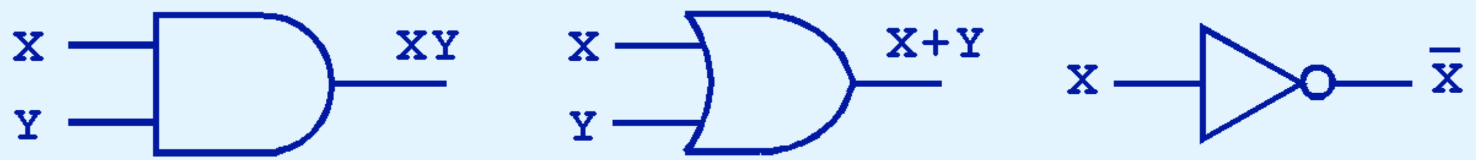
X	Y	Z	*f*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Logic Gates

- We have looked at Boolean functions in abstract terms.
- In this section, we see that Boolean functions are implemented in digital computer circuits called **gates**.
- **A gate is an electronic device that produces a result based on two or more input values.**
  - **In reality**, **gates consist of one to six transistors**, but digital designers think of them as a single unit.
  - Integrated circuits contain collections of gates suited to a particular purpose.

Logic Gates – AND OR NOT

- The three simplest gates are the AND, OR, and NOT gates.



X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

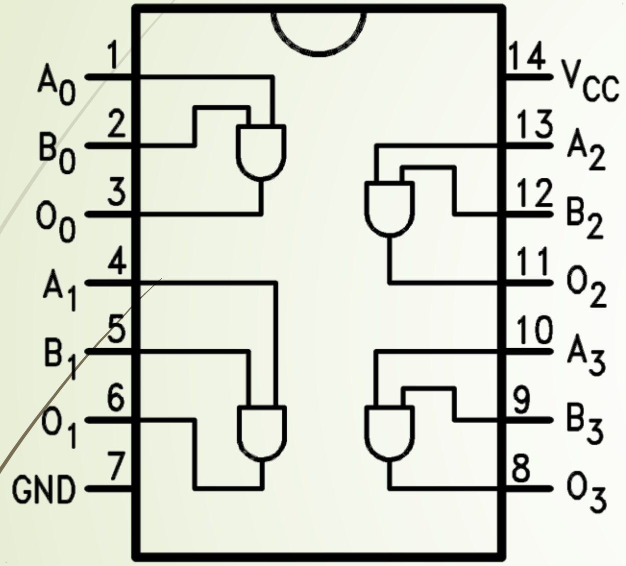
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT X

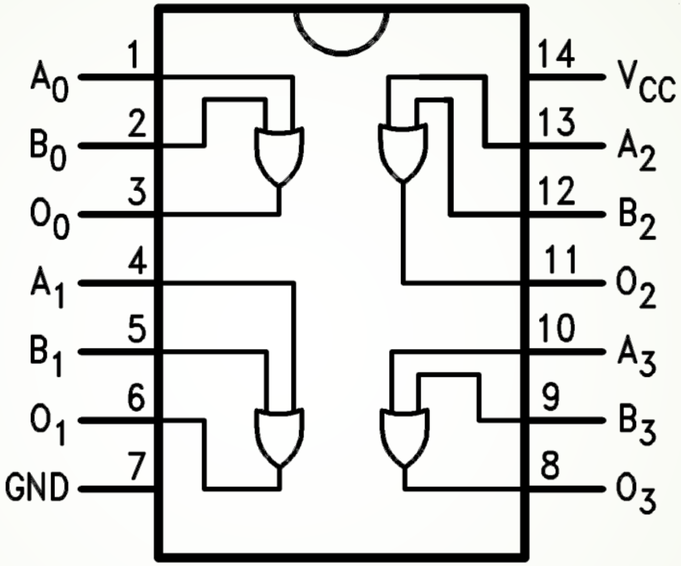
X	$\overline{X}$
0	1
1	0

- They correspond directly to their respective Boolean operations, as you can see by their truth tables.

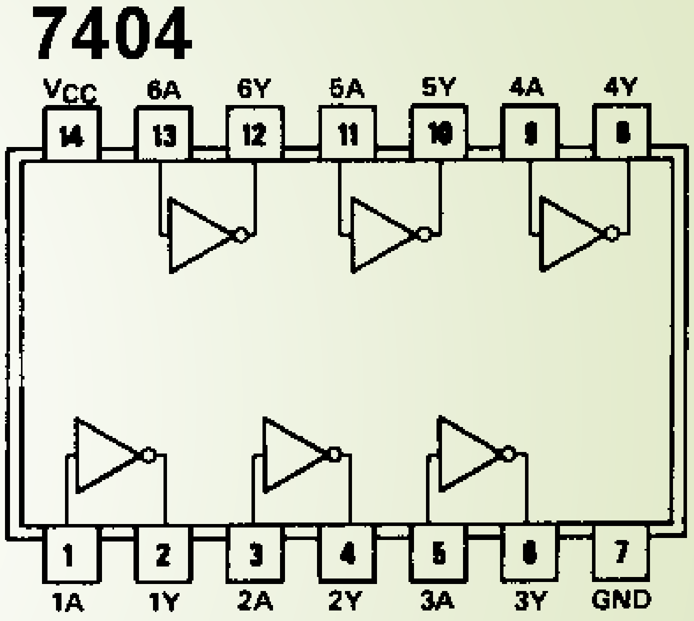
Logic Gates – AND OR NOT



74AC08, 74ACT08
Quad 2-Input AND Gate



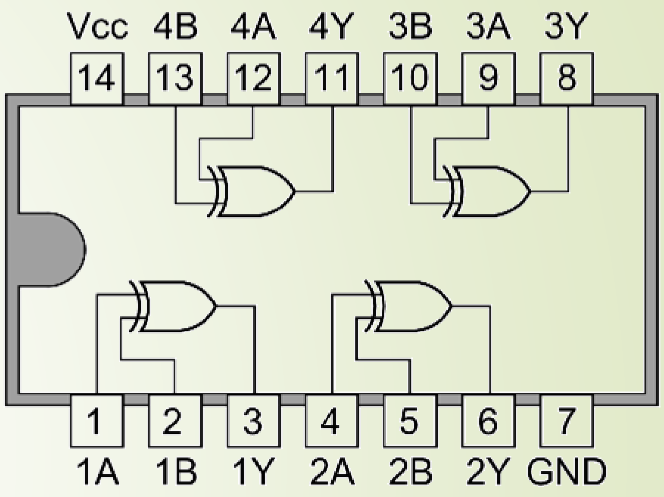
74AC32, 74ACT32
Quad 2-Input OR Gate



7404
6-Input NOT Gate

Logic Gates – XOR

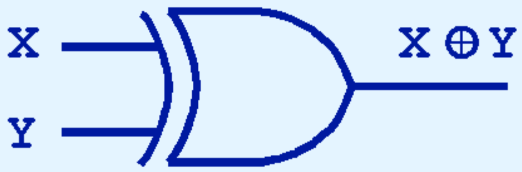
- Another very useful gate is the exclusive OR (XOR) gate.
- The output of the XOR operation is true only when the values of the inputs differ.



74HCT86 Quad 2-input EXCLUSIVE-OR gate

X XOR Y

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0



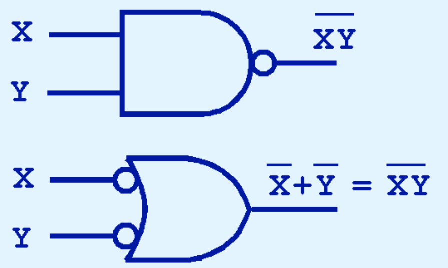
Note the special symbol ⊕ for the XOR operation.

Logic Gates – NAND NOR

- NAND and NOR are two very important gates.
- Their symbols and truth tables are shown at the right.

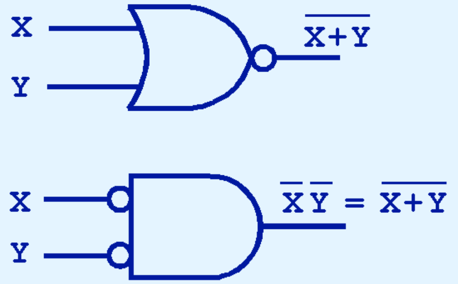
X NAND Y

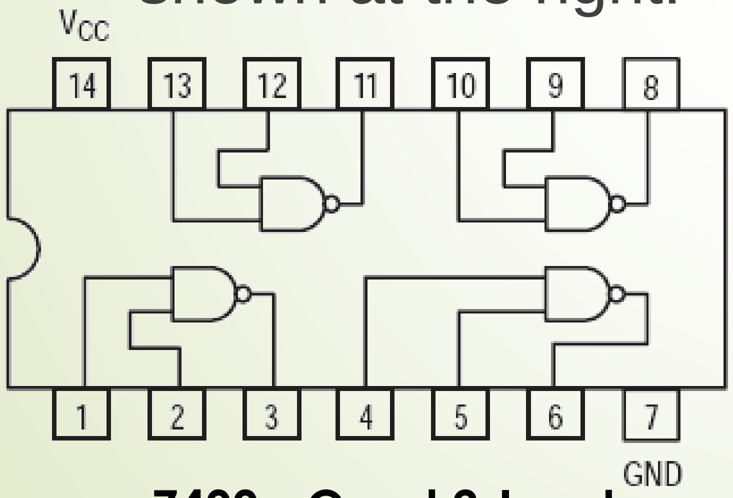
X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0



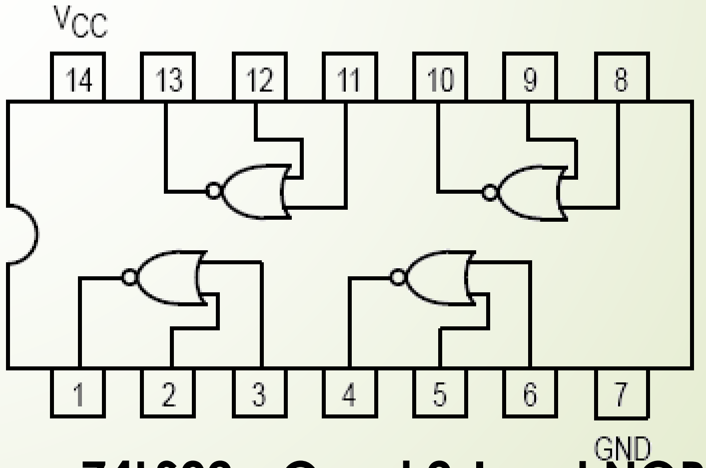
X NOR Y

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0





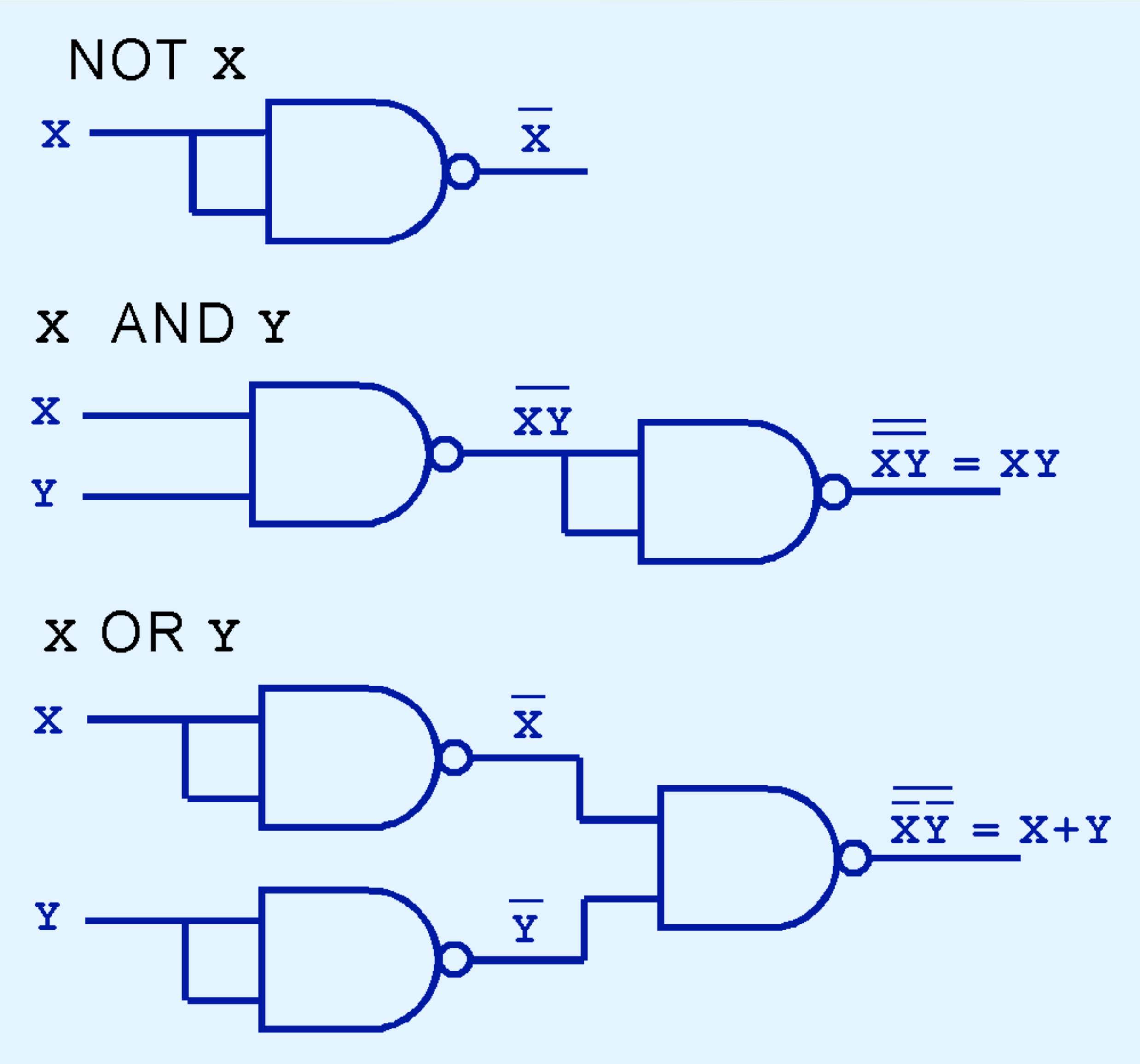
7400 - Quad 2-Input NAND Gate



74LS02 - Quad 2-Input NOR Gate

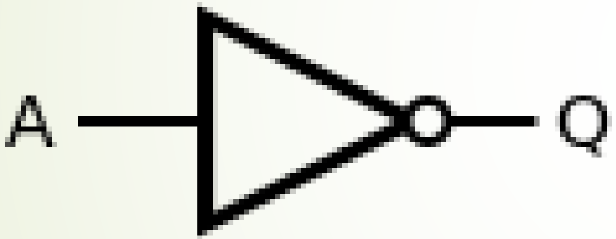
Logic Gates

- NAND and NOR are known as ***universal gates*** because they are **inexpensive** to manufacture and **any Boolean function can be constructed using only NAND or only NOR gates.**



Logic Gates NAND with NOT gate

- Target

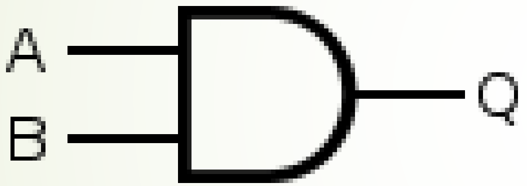


NAND usage

Truth Table	
Input A	Output Q
0	1
1	0

Logic Gates NAND with AND

- Target



NAND usage

Truth Table		
Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Logic Gates NAND with OR

OR gate from NAND gates

NAND (Not AND)		
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

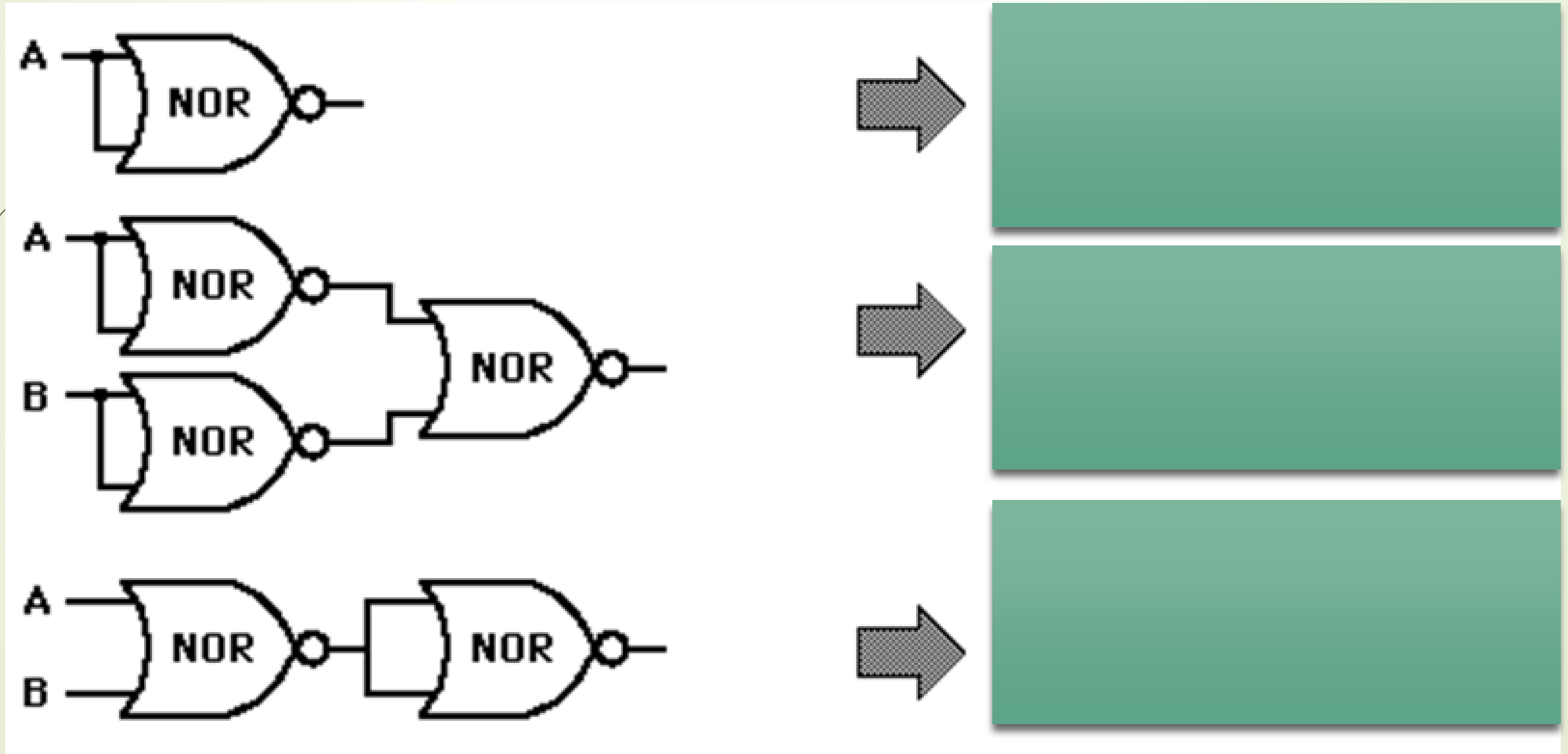
Logic Gates NAND with NOR Gate

NOR gate from NAND gates

NAND (Not AND)		
A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

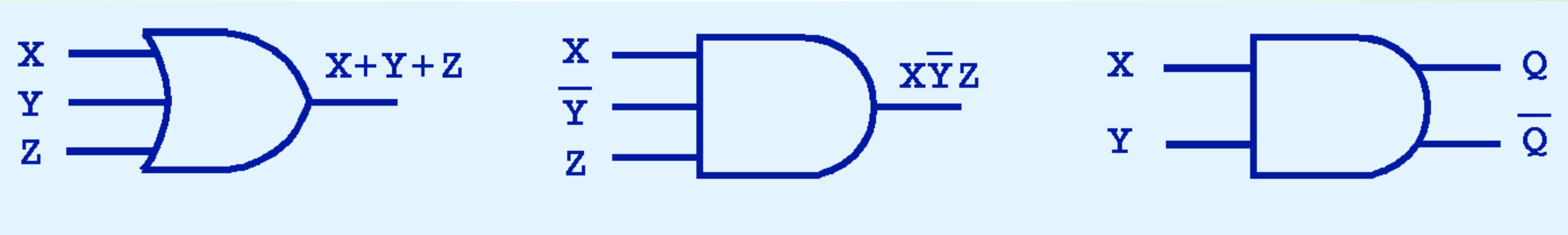
NOR		
A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Logic Gates – Creating other gates using NOR gate



Logic Gates

- Gates can have multiple inputs and more than one output.
  - A second output can be provided for the complement of the operation.
  - We'll see more of this later.



Logic Gates – Example 1

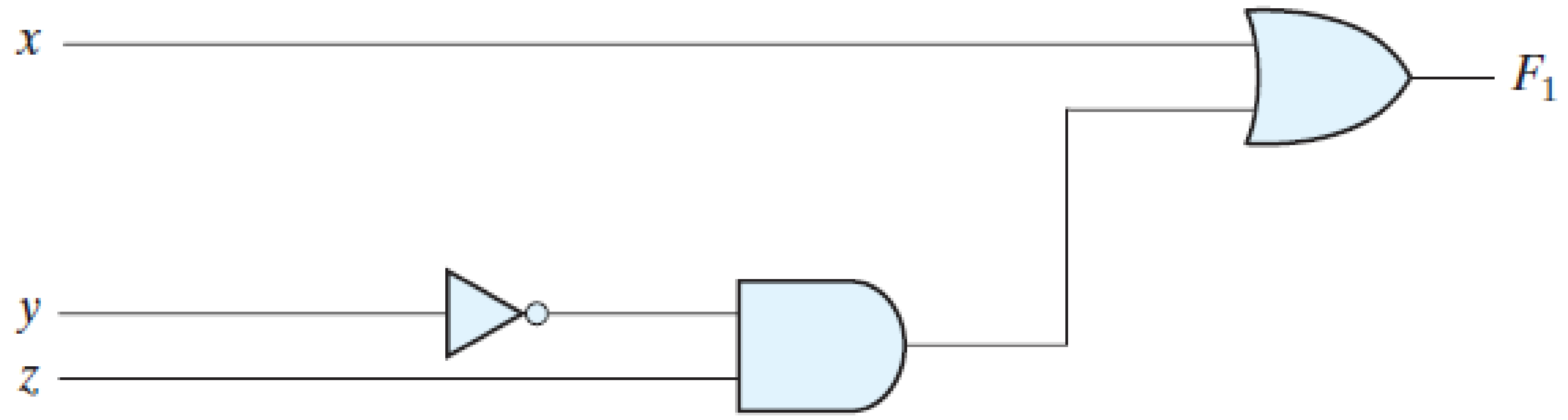
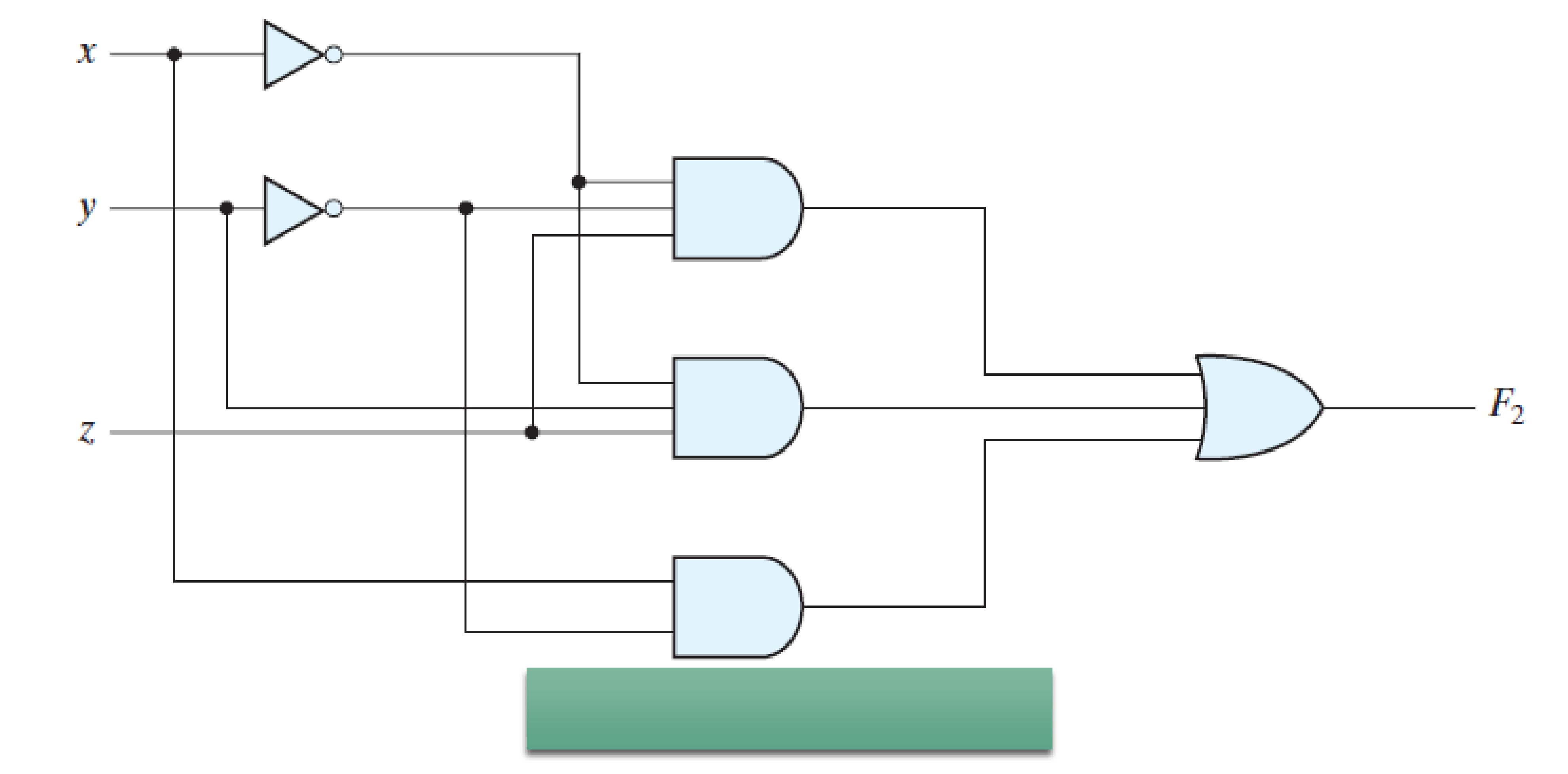


FIGURE 2.1
Gate implementation of $F_1 = x + y'z$

Logic Gates – Example 2



Logic Gates – Example 3